# The Monte Carlo Database System for Querying Imprecise, Uncertain, and Missing Data

Chris Jermaine
Rice University

Current/Recent MCDB/SimSQL team: Zhuhua Cai, Jacob Gao, Michael Gubanov, Shangyu Luo, Luis Perez
Also, Peter J. Haas at IBM Almaden

# Today: Will Talk About **MCDB/SimSQL**

# Today: Will Talk About **MCDB/SimSQL**

...And about “Stochastic Analytics”...

# What Is **MCDB/SimSQL?**

- Database system developed over many years at Rice

# What Is **MCDB/SimSQL?**

- Database system developed over many years at Rice
- Lives in the Hadoop Ecosystem

“The Monte Carlo Database System”

# What Is **MCDB/SimSQL?**

- Database system developed over many years at Rice
- Lives in the Hadoop Ecosystem

“The Monte Carlo Database System”

- First and foremost, it is an SQL database

# What Is **MCDB/SimSQL?**

- Database system developed over many years at Rice
- Lives in the Hadoop Ecosystem

  "The Monte Carlo Database System"

- First and foremost, it is an SQL database
- But it is unique in its native support for **stochastic analytics**

# What Is Meant by *Stochastic Analytics*?

# What Is Meant by *Stochastic Analytics*?

- Tackling (Big Data) analytic tasks using stochastic models

# First: What Is a *Stochastic Model*?

- A model for some slice of reality which has a *random* component
  - — Here *random* means *probabilistic*
  - — There is typically a distribution over possible inputs and/or outcomes
- Why utilize randomness?

# First: What Is a *Stochastic Model*?

- A model for some slice of reality which has a *random* component

  — Here *random* means *probabilistic*

  — There is typically a distribution over possible inputs and/or outcomes

- Why utilize randomness?

  — Randomness provides a way to model uncertainty

  Have a data record describing “J. Smith”
  Both “John Smith” and “Jane Smith” are people in your data set
  Record a 50/50 chance of “J. Smith” referring to John/Jane

# First: What Is a *Stochastic Model*?

- A model for some slice of reality which has a *random* component
  - — Here *random* means *probabilistic*
  - — There is typically a distribution over possible inputs and/or outcomes
- Why utilize randomness?
  - — Randomness provides a way to model uncertainty
  - — Provides a way to model missing data

    Missing a person's gender?
    52% of people in the data set are women...
    Represent the gender via a distribution (52% female, 48% male)

# First: What Is a *Stochastic Model*?

- A model for some slice of reality which has a *random* component
  - — Here *random* means *probabilistic*
  - — There is typically a distribution over possible inputs and/or outcomes
- Why utilize randomness?
  - — Randomness provides a way to model uncertainty
  - — Provides a way to model missing data
  - — Provides for a principled way to talk about beliefs

    "I am 50% sure that this is gonna be a great presentation!"

# First: What Is a *Stochastic Model*?

- A model for some slice of reality which has a *random* component
  - — Here *random* means *probabilistic*
  - — There is typically a distribution over possible inputs and/or outcomes
- Why utilize randomness?
  - — Randomness provides a way to model uncertainty
  - — Provides a way to model missing data
  - — Provides for a principled way to talk about beliefs
- One great thing about randomness:
  - — The theory already exists

    Can leverage 100s of years of probability theory

# Now: What About *Stochastic Analytics*?

Application of stochastic models/methods to Big Data analytics

# Now: What About *Stochastic Analytics*?

Application of stochastic models/methods to Big Data analytics

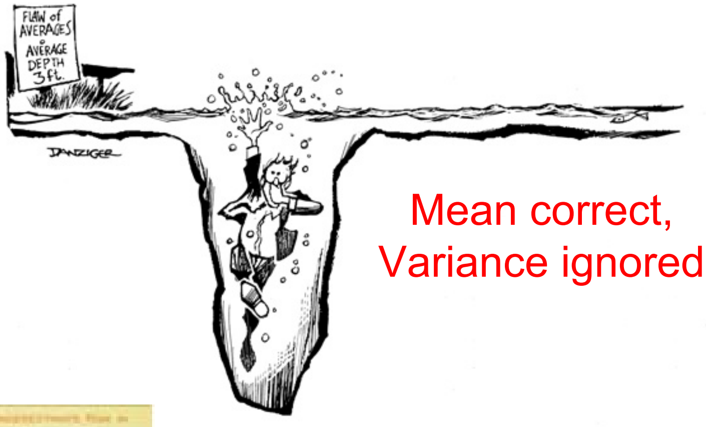
- With the wrinkle that the result of the analysis is stochastic
- Gives analyst an idea of **risk/uncertainty** of result
- Or an idea of the **risk/uncertainty** of the modelling assumptions

# Why Should I Care About Risk/Uncertainty?

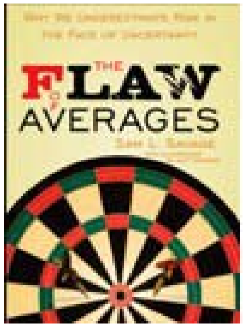
# Why Should I Care About Risk/Uncertainty?

- Worthwhile polemic is Sam Savage (Stanford) *Flaw of Averages*
- Savage describes two “f/laws”:

Flaw of averages (weak form):

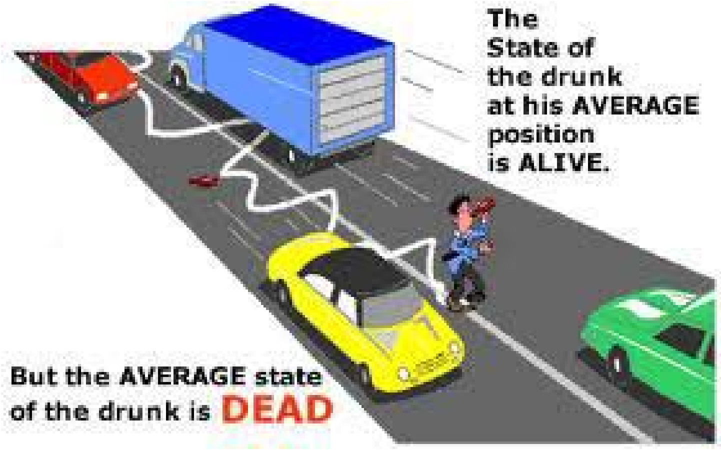


Mean correct,
Variance ignored



Sam Savage’s book

Flaw of averages (strong form):


The State of the drunk at his AVERAGE position is ALIVE.
But the AVERAGE state of the drunk is DEAD

Wrong value of mean:
$f(E[X]) \neq E[f(X)]$

# So How Does MCDB/SimSQL Help?

- Makes it easy to attach fine-grained “what if” models to database
- Or to attach models to unknown/uncertain data
- Queries to models look like any other query
- Except that they get back a **distribution** of results

This distribution encodes the uncertainty in the analysis

# What’s The Significance of “Fine-Grained”?

- Just imagine...
- We have archived billions of sales records and want to know:

  “What would my profits have been in ‘14 if I’d cut all of my margins by 10%?”

- Classical approach: a single, aggregate model
  - Problem: Typically under/over-estimates variance
  - What if you have a few, high-margin items where demand is inelastic?
- Instead:
  - Dive deep, model each customer, perhaps each purchase

# How Do I Make a Model Stochastic?

- ...So it emits a distribution of results?
- Go **Bayesian**

# The Bayesian Approach

- Come up with a stochastic generative process
- Which includes prior distributions on the quantities like to est
  - — In our example, a prior on the demand curve
- See some data
- Use Bayes' Theorem and data to “update” the priors
  - — This gives you a posterior dist
  - — The posterior is your estimate
- Why attractive for use in Big Data analytics?
  - — Answer is a dist
  - — So it gives you some idea of uncertainty/risk of inferences made using the data

# Now Back to the Example

- We have archived billions of sales records and want to know:

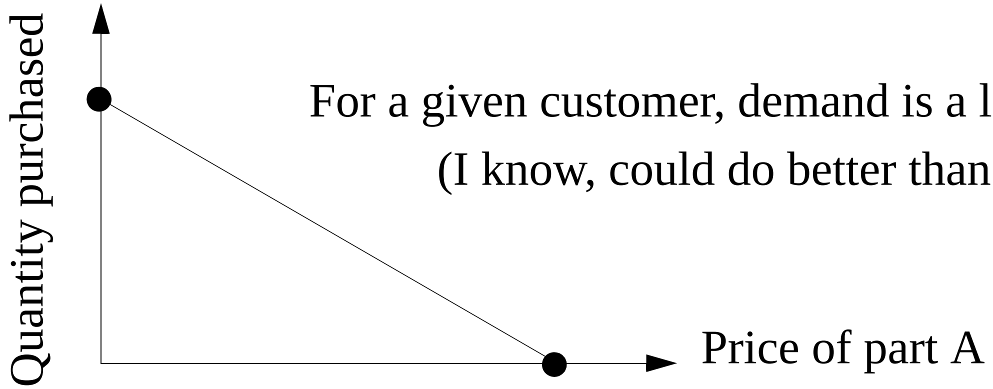
  “What would my profits have been in ‘14 if I’d cut all of my margins by 10%?”

- Here’s one possibility...

  — ...utilizing the Bayesian approach

# Step 1: A Stochastic Demand Model

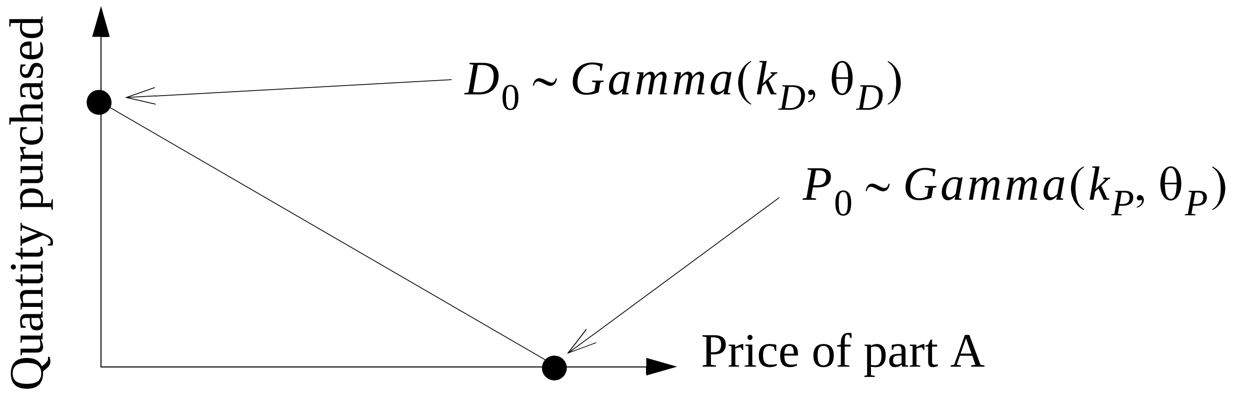
- First, define a customer demand model...



For a given customer, demand is a linear function
(I know, could do better than linear...)

# Step 1: A Stochastic Demand Model

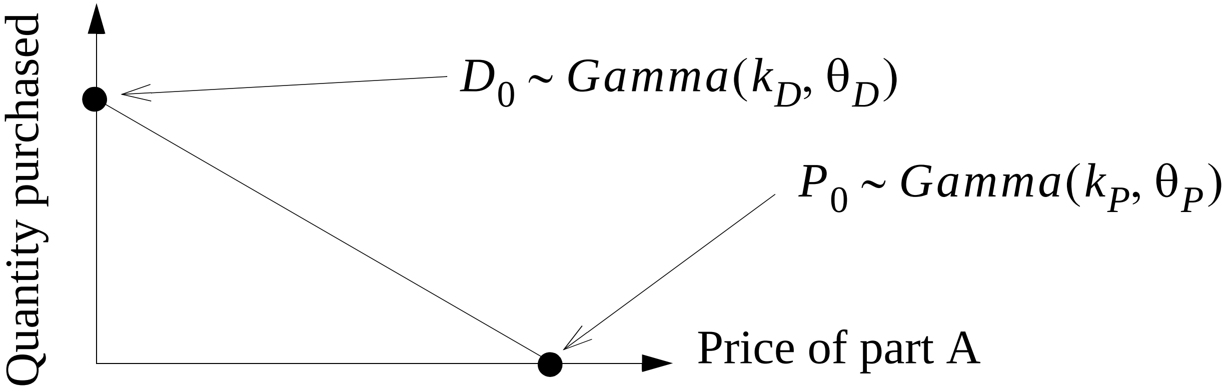
- First, define a customer demand model...


Quantity purchased
$D_0 \sim Gamma(k_D, \theta_D)$
$P_0 \sim Gamma(k_P, \theta_P)$
Price of part A

Demand curve is generated via samples from twin Gamma distributions

# Step 1: A Stochastic Demand Model

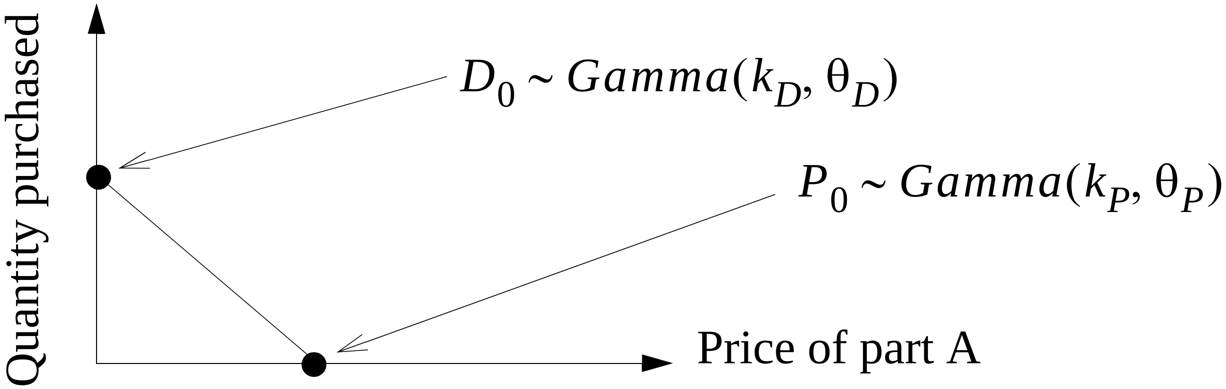
- First, define a customer demand model...


$D_0 \sim Gamma(k_D, \theta_D)$
$P_0 \sim Gamma(k_P, \theta_P)$
Quantity purchased
Price of part A

Distribution over $D_0$, $P_0$ defines a whole family of possible demand curves...

# Step 1: A Stochastic Demand Model

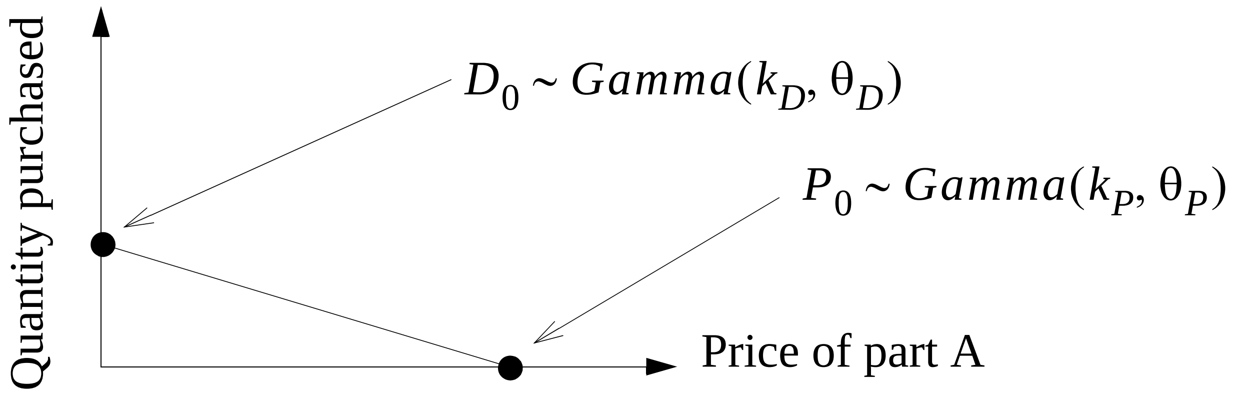
- First, define a customer demand model...


Quantity purchased
$D_0 \sim Gamma(k_D, \theta_D)$
$P_0 \sim Gamma(k_P, \theta_P)$
Price of part A

Here’s a new, possible demand curve...

# Step 1: A Stochastic Demand Model

- First, define a customer demand model...

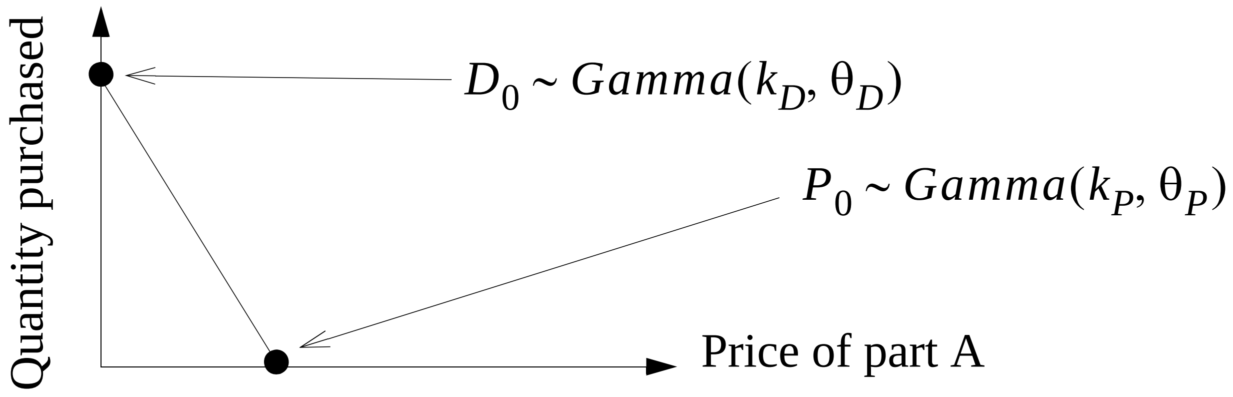


Quantity purchased
$D_0 \sim Gamma(k_D, \theta_D)$
$P_0 \sim Gamma(k_P, \theta_P)$
Price of part A

And another...

# Step 1: A Stochastic Demand Model

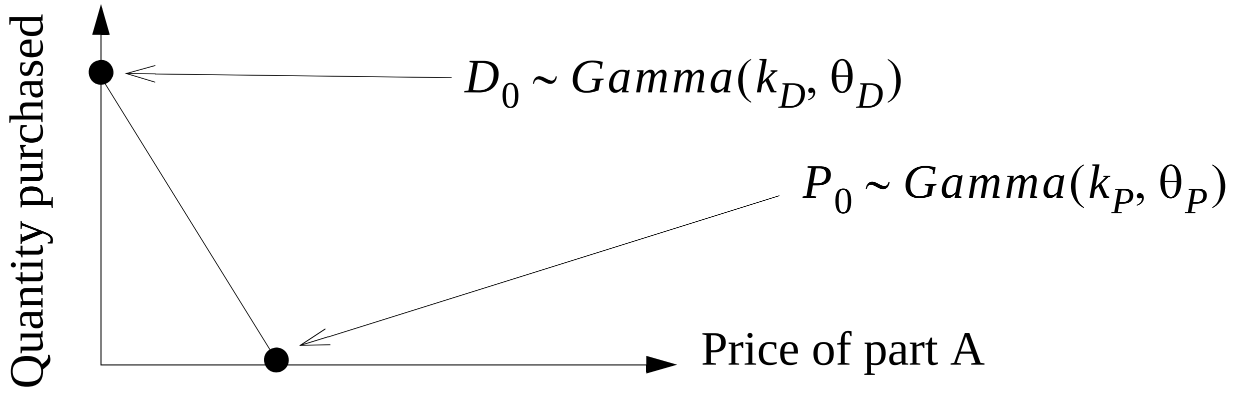
- First, define a customer demand model...


$D_0 \sim Gamma(k_D, \theta_D)$
$P_0 \sim Gamma(k_P, \theta_P)$
Quantity purchased
Price of part A

And so on...

# Step 1: A Stochastic Demand Model

- First, define a customer demand model...


Quantity purchased
$D_0 \sim Gamma(k_D, \theta_D)$
$P_0 \sim Gamma(k_P, \theta_P)$
Price of part A

- The PDF

$$F(f\ |.)\ =\ \mathrm{Gamma}(D_0|k_D, \theta_D) \times \mathrm{Gamma}(P_0|k_P, \theta_P)$$

is our prior distribution over demand curves

# Step 2: “Learn” the Model

- To apply this model, we need to “learn” the prior
  - — That is, choose $k_P$, $k_D$, $\theta_P$, $\theta_D$ to be realistic for sales in 2014
- Reasonable tactic: use the warehoused data to perform an MLE
  - — That is, find the set of params that best describes all of the 2008 sales
  - — Do this by issuing computations over the warehouse

# Step 3: Apply the Model - the Theory

- Now we have a prior model (PDF) for demand function *f*:

$$F(f \mid k_P, k_D, \theta_P, \theta_D)$$

- Problem: the actual demand curve for each customer is unseen
- But can use observed demand to obtain a *posterior* demand model
- Ex: for *i*th sale, a customer bought *d* units at price *p*
- Then posterior demand model is given by:

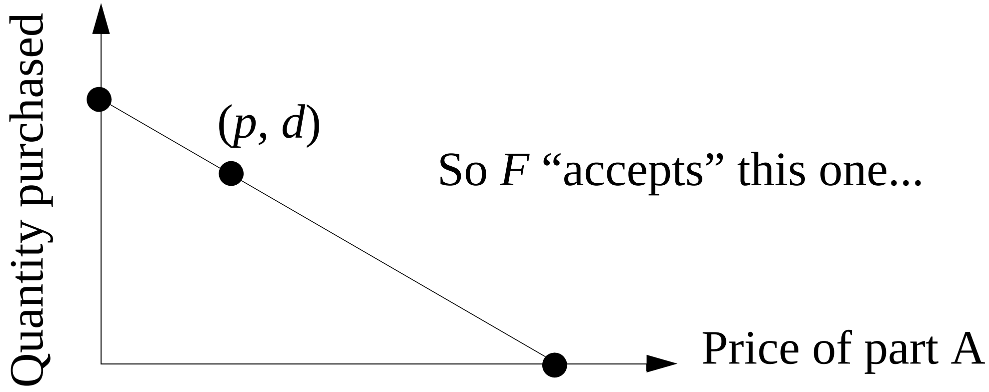
$$F(f_i \mid k_P, k_D, \theta_P, \theta_D, f_i(p) = d)$$

# Step 3: Apply the Model - the Theory

- Intuitively, $F(f_i \mid f_i(p) = d)$ gives non-zero “weight” to all demand functions thru the point $(p, d)$

# Step 3: Apply the Model - the Theory

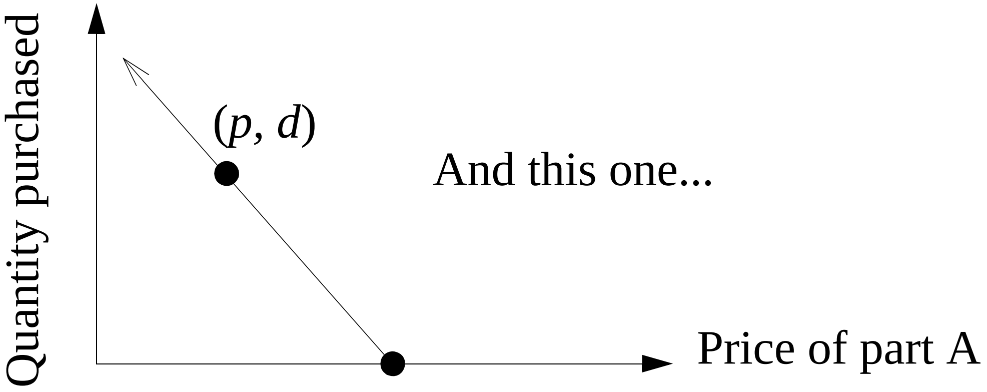
- Intuitively, $F(f_i \mid f_i(p) = d)$ gives non-zero “weight” to all demand functions thru the point ($p$, $d$)



So $F$ “accepts” this one...

# Step 3: Apply the Model - the Theory

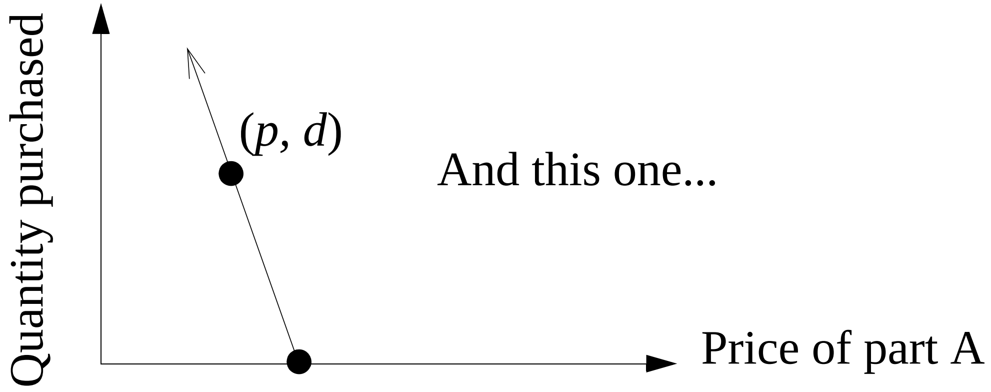
- Intuitively, $F(f_i \mid f_i(p) = d)$ gives non-zero "weight" to all demand functions thru the point $(p, d)$



And this one...

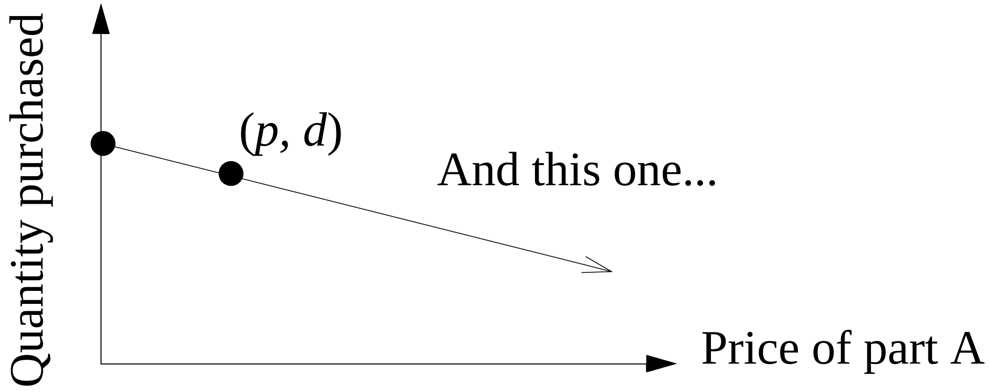
# Step 3: Apply the Model - the Theory

- Intuitively, $F(f_i \mid f_i(p) = d)$ gives non-zero "weight" to all demand functions thru the point $(p, d)$



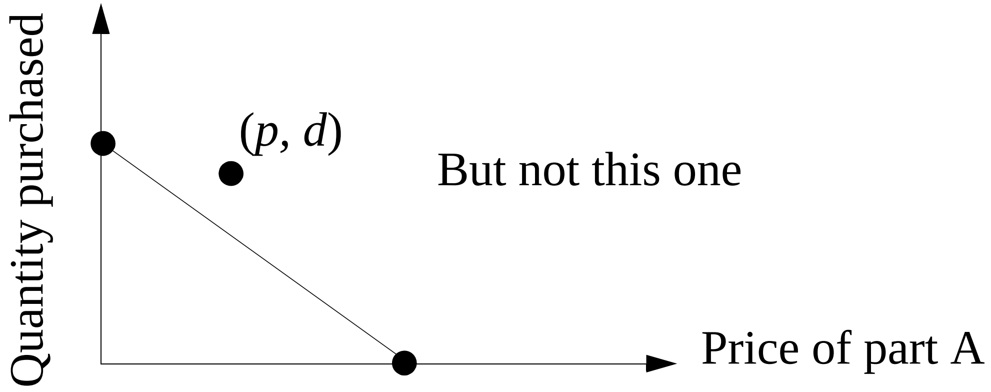
# Step 3: Apply the Model - the Theory

- Intuitively, $F(f_i \mid f_i(p) = d)$ gives non-zero “weight” to all demand functions thru the point $(p, d)$



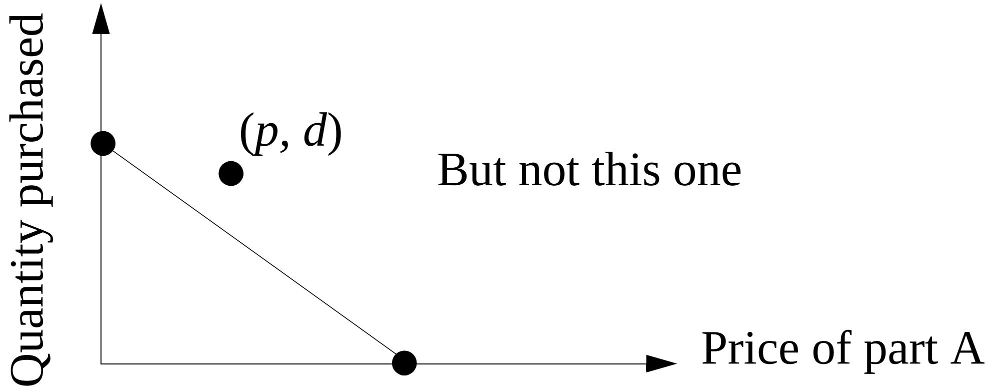
# Step 3: Apply the Model - the Theory

- Intuitively, $F(f_i \mid f_i(p) = d)$ gives non-zero “weight” to all demand functions thru the point $(p, d)$



# Step 3: Apply the Model - the Theory

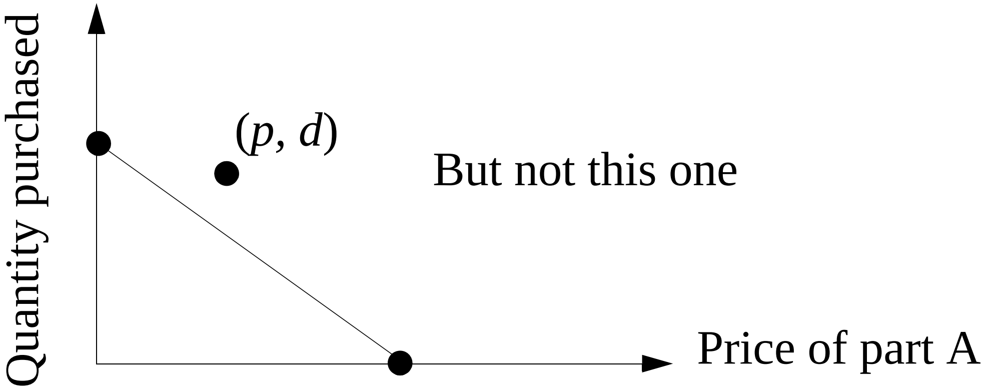
- Intuitively, $F(f_i \mid f_i(p) = d)$ gives non-zero “weight” to all demand functions thru the point $(p, d)$



- Those demand curves that are given non-zero weight are ordered exactly as the prior would order them

# Step 3: Apply the Model - the Theory

- Intuitively, $F(f_i \mid f_i(p) = d)$ gives non-zero "weight" to all demand functions thru the point $(p, d)$



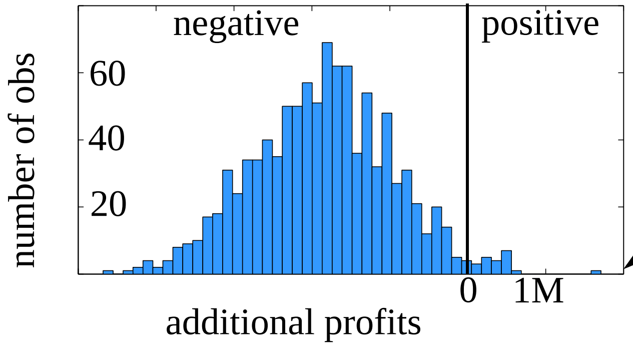
- Those demand curves that are given non-zero weight are ordered exactly as the prior would order them
- To "guess" the customer's demand at new price $p'$:
  - — Sample a possible $f_i$ from the PDF $F(f_i \mid k_P, k_D, \theta_P, \theta_D, f_i(p) = d)$
  - — Compute $f_i(p')$, and we have a new demand!

# Step 3: Apply the Model - the Practice

- To actually **compute** the overall profit under new prices:
  - — I’d need to sample an $f_i$ from $F(f_i \mid f_i(p) = d)$ for every sale from 2014
  - — Evaluate $f_i(p')$ for all those sales
  - — Compute the new profits

# Step 3: Apply the Model - the Practice

- Issue: the profits are actually random
  - — You do this once, you get one answer
  - — You do this again, you get another answer
  - — How to handle this?
  - — Redo the computation many times (Monte Carlo) to obtain a *distribution* of results



don’t do it!

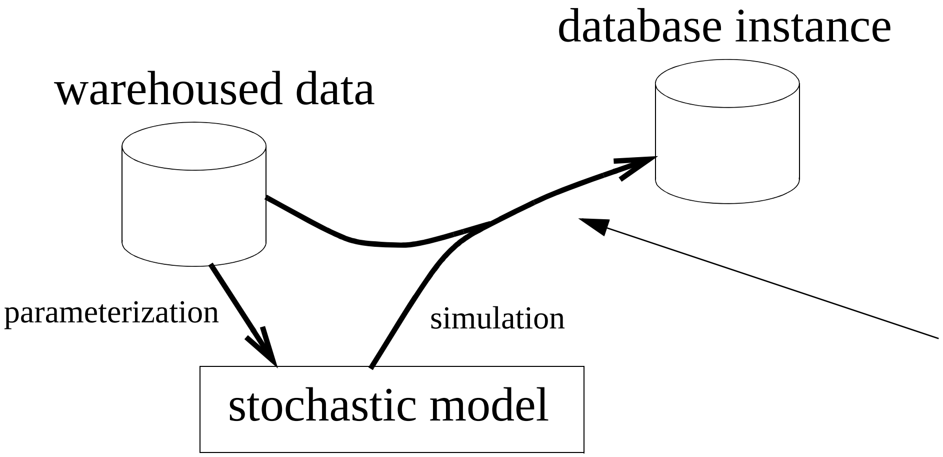
MCDB: you supply the model,
you ask the question,
it handles the stochastic part

# Illustrative of Stochastic Analytics

- Began with a stochastic model
- Model was applied at very fine granularity to big data set
- Result of analysis was a distribution, not a single result

# MCDB Makes This Process Easy

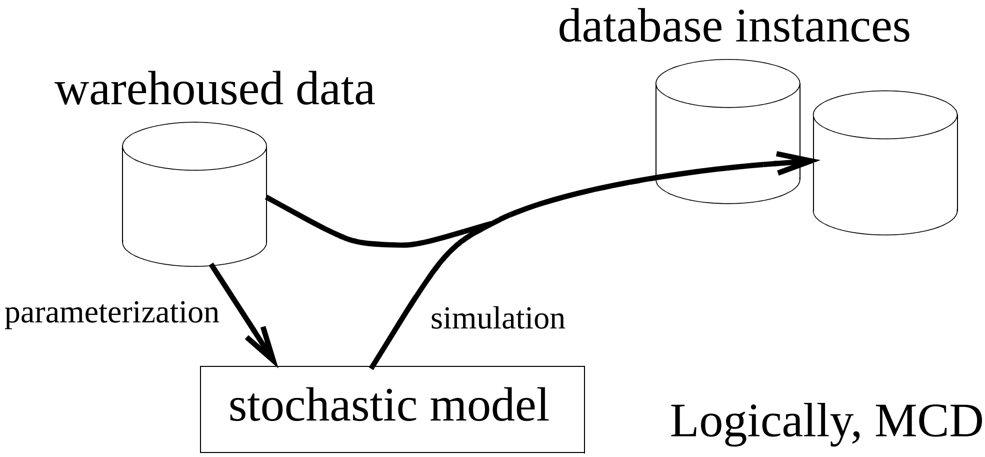
- In MCDB, easy to associate a posterior dist of demand curves...
  - — with every one of the 100M customers in a large database
  - — And then use those curves to generate stochastic DB instances



Done by implementing a “VG Function” which performs the simulation

# This Is Where MCDB Comes In

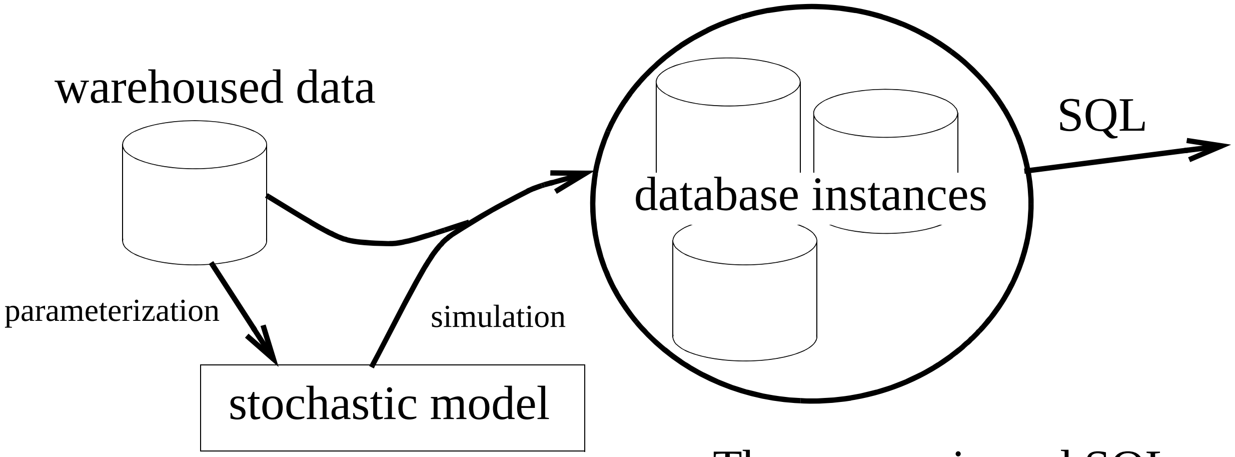
- In MCDB, easy to associate a posterior dist of demand curves...
  - — with every one of the 100M customers in a large database
  - — And then use those curves to generate stochastic DB instances



Logically, MCDB generates many database instances (“possible worlds”)

# This Is Where MCDB Comes In

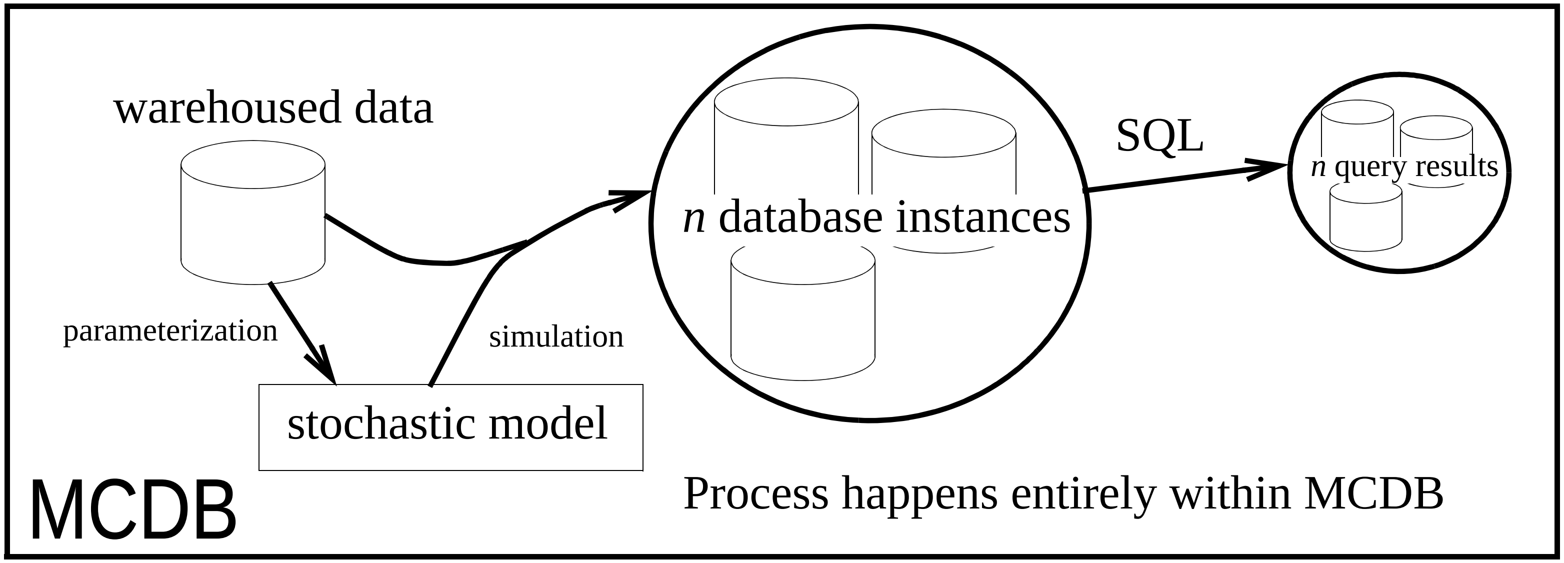
- In MCDB, easy to associate a posterior dist of demand curves...
  - — with every one of the 100M customers in a large database
  - — And then use those curves to generate stochastic DB instances



Then a user-issued SQL query
is simultaneously evaluated over all instances

# This Is Where MCDB Comes In

- In MCDB, easy to associate a posterior dist of demand curves...
  - — with every one of the 100M customers in a large database
  - — And then use those curves to generate stochastic DB instances



# What Does MCDB Look Like to a Programmer?

# MCDB's Version of SQL

- Most fundamental SQL addition is "VG Function" abstraction
  - — Many built-in models
  - — But via UDF interface, can support just about anything (simple statistical distributions, complex Bayesian models, graphical models, neural models, etc.)

# MCDB’s Version of SQL

- Most fundamental SQL addition is “VG Function” abstraction
- Called via a special `CREATE TABLE` statement
- Example; assuming:
  - `SBP(MEAN, STD, GENDER)`
  - `PATIENTS(NAME, GENDER)`
- To create a random table, we might have:

```
CREATE TABLE SBP_DATA(NAME, GENDER, SBP) AS
FOR EACH p in PATIENTS
 WITH Res AS Normal (
  SELECT s.MEAN, s.STD
  FROM SPB s WHERE s.GENDER = p.GENDER)
 SELECT p.NAME, p.GENDER, r.VALUE
 FROM Res r
```

Sorry for the code.
After all, I’m a
computer scientist!

# How Does This Work?

```
CREATE TABLE SBP_DATA(NAME, GENDER, SBP) AS
FOR EACH p in PATIENTS
  WITH Res AS Normal (
    SELECT s.MEAN, s.STD
    FROM SPB s WHERE s.GENDER = p.GENDER)
  SELECT p.NAME, p.GENDER, r.VALUE
  FROM Res r
```

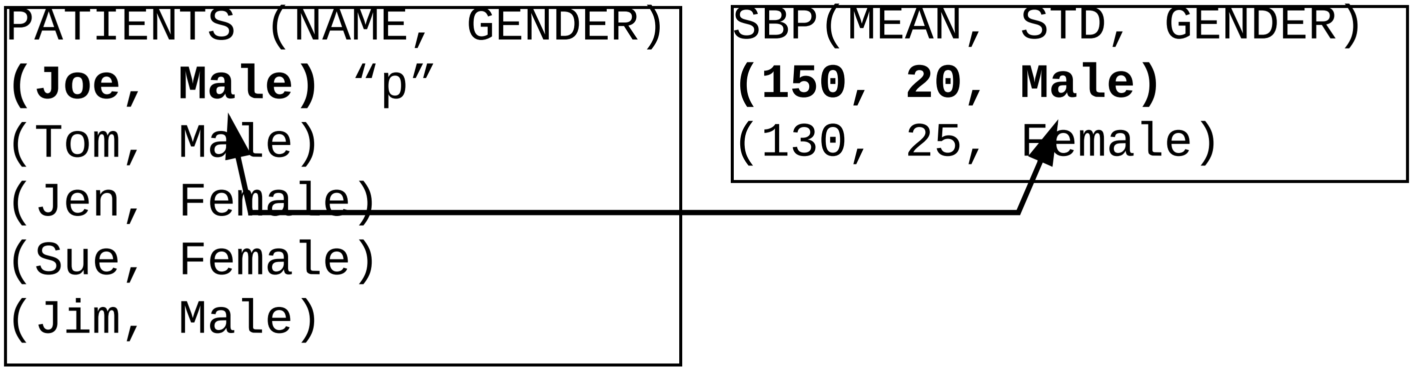
Loop through PATIENTS

```
PATIENTS (NAME, GENDER)
(Joe, Male) “p”
(Tom, Male)
(Jen, Female)
(Sue, Female)
(Jim, Male)
```

```
SBP(MEAN, STD, GENDER)
(150, 20, Male)
(130, 25, Female)
```

# How Does This Work?

```
CREATE TABLE SBP_DATA(NAME, GENDER, SBP) AS
FOR EACH p in PATIENTS
  WITH Res AS Normal (
    SELECT s.MEAN, s.STD
    FROM SPB s WHERE s.GENDER = p.GENDER)
  SELECT p.NAME, p.GENDER, r.VALUE
  FROM Res r
```

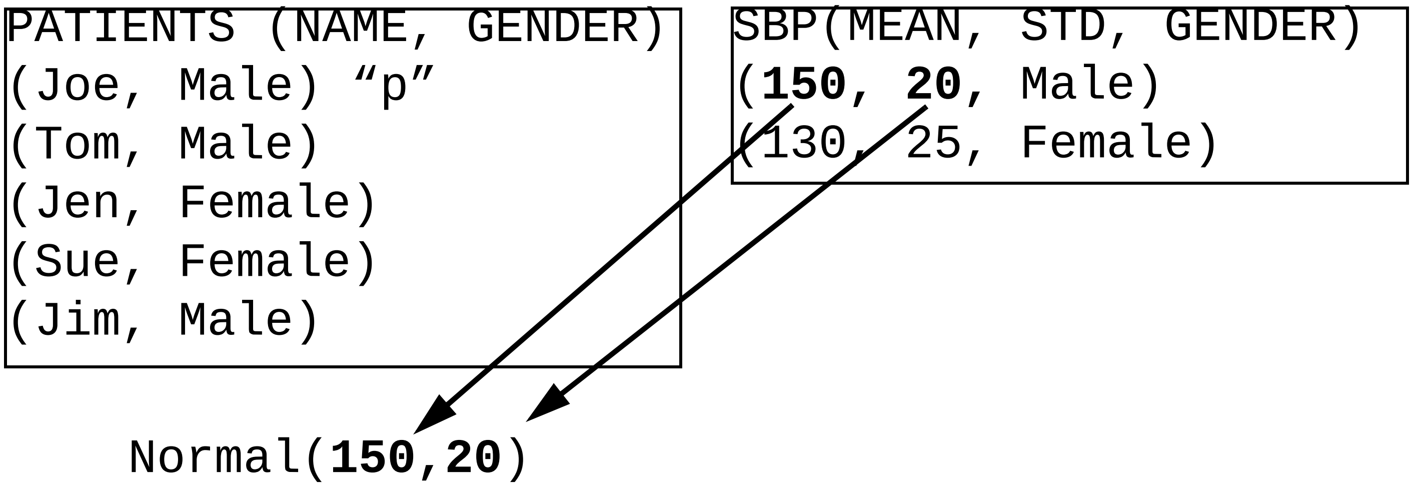


```
PATIENTS (NAME, GENDER)
(Joe, Male) “p”
(Tom, Male)
(Jen, Female)
(Sue, Female)
(Jim, Male)
```

```
SBP(MEAN, STD, GENDER)
(150, 20, Male)
(130, 25, Female)
```

# How Does This Work?

```
CREATE TABLE SBP_DATA(NAME, GENDER, SBP) AS
FOR EACH p in PATIENTS
  WITH Res AS Normal (
    SELECT s.MEAN, s.STD
    FROM SPB s WHERE s.GENDER = p.GENDER)
  SELECT p.NAME, p.GENDER, r.VALUE
  FROM Res r
```



PATIENTS (NAME, GENDER)
(Joe, Male) “p”
(Tom, Male)
(Jen, Female)
(Sue, Female)
(Jim, Male)

SBP(MEAN, STD, GENDER)
(**150**, **20**, Male)
(130, 25, Female)

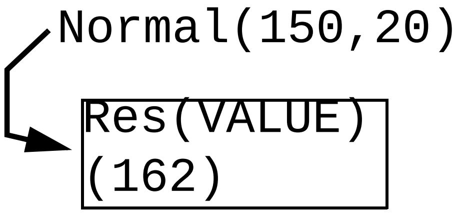
Normal(**150**,**20**)

# How Does This Work?

```
CREATE TABLE SBP_DATA(NAME, GENDER, SBP) AS
FOR EACH p in PATIENTS
  WITH Res AS Normal (
    SELECT s.MEAN, s.STD
    FROM SPB s WHERE s.GENDER = p.GENDER)
  SELECT p.NAME, p.GENDER, r.VALUE
  FROM Res r
```

```
PATIENTS (NAME, GENDER)
(Joe, Male) “p”
(Tom, Male)
(Jen, Female)
(Sue, Female)
(Jim, Male)
```

```
SBP(MEAN, STD, GENDER)
(150, 20, Male)
(130, 25, Female)
```

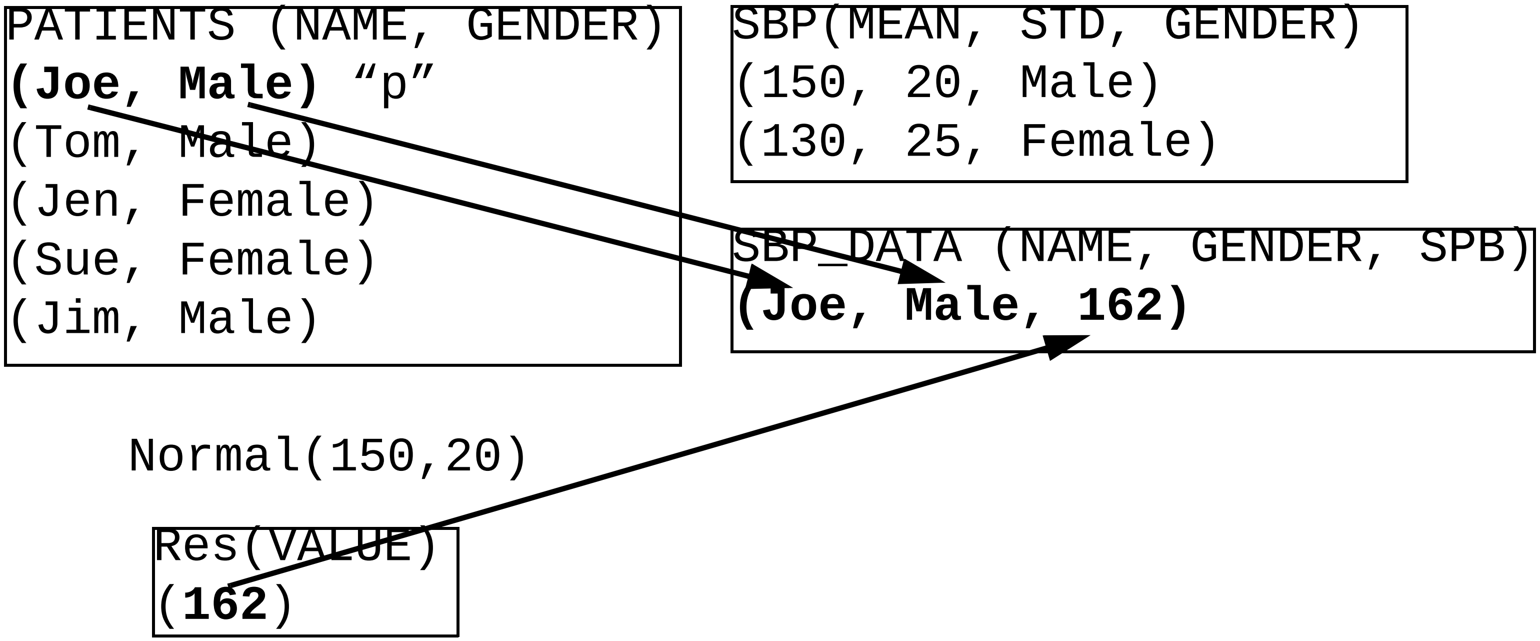


```
Normal(150,20)

Res(VALUE)
(162)
```

# How Does This Work?

```
CREATE TABLE SBP_DATA(NAME, GENDER, SBP) AS
FOR EACH p in PATIENTS
  WITH Res AS Normal (
    SELECT s.MEAN, s.STD
    FROM SPB s WHERE s.GENDER = p.GENDER)
  SELECT p.NAME, p.GENDER, r.VALUE
  FROM Res r
```



PATIENTS (NAME, GENDER)
**(Joe, Male)** “p”
(Tom, Male)
(Jen, Female)
(Sue, Female)
(Jim, Male)

SBP(MEAN, STD, GENDER)
(150, 20, Male)
(130, 25, Female)

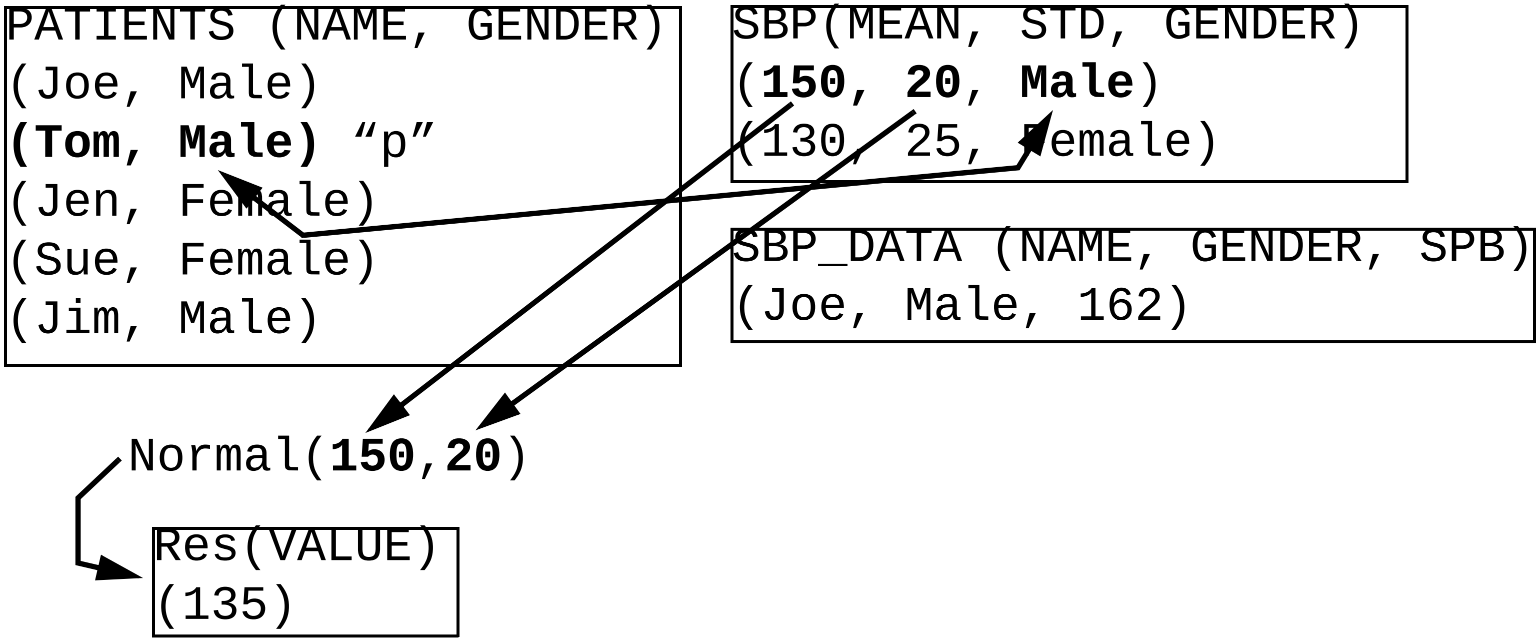
SBP_DATA (NAME, GENDER, SPB)
**(Joe, Male, 162)**

Normal(150,20)

Res(VALUE)
(**162**)

# How Does This Work?

```
CREATE TABLE SBP_DATA(NAME, GENDER, SBP) AS
FOR EACH p in PATIENTS
  WITH Res AS Normal (
    SELECT s.MEAN, s.STD
    FROM SPB s WHERE s.GENDER = p.GENDER)
  SELECT p.NAME, p.GENDER, r.VALUE
  FROM Res r
```



```
PATIENTS (NAME, GENDER)
(Joe, Male)
(Tom, Male) “p”
(Jen, Female)
(Sue, Female)
(Jim, Male)
```

```
SBP(MEAN, STD, GENDER)
(150, 20, Male)
(130, 25, Female)
```

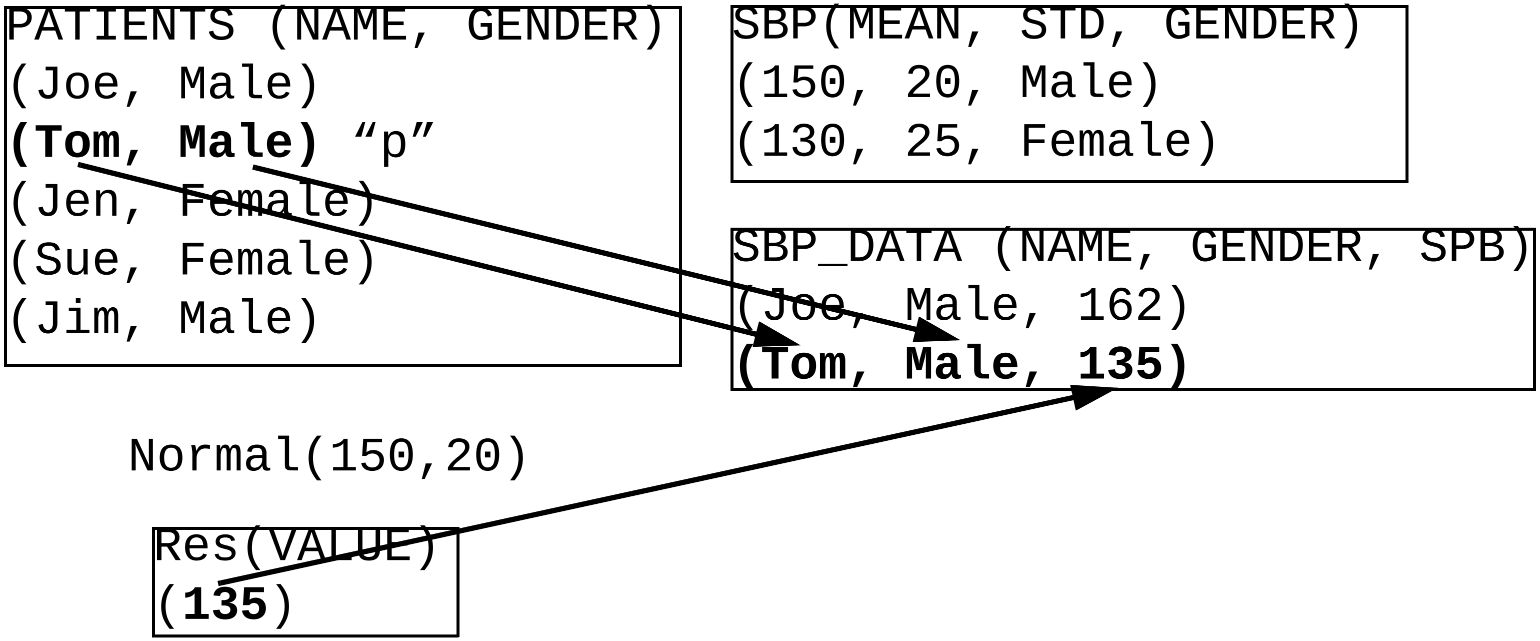
```
SBP_DATA (NAME, GENDER, SPB)
(Joe, Male, 162)
```

```
Normal(150,20)
```

```
Res(VALUE)
(135)
```

# How Does This Work?

```
CREATE TABLE SBP_DATA(NAME, GENDER, SBP) AS
FOR EACH p in PATIENTS
  WITH Res AS Normal (
    SELECT s.MEAN, s.STD
    FROM SPB s WHERE s.GENDER = p.GENDER)
  SELECT p.NAME, p.GENDER, r.VALUE
  FROM Res r
```



```
PATIENTS (NAME, GENDER)
(Joe, Male)
(Tom, Male) “p”
(Jen, Female)
(Sue, Female)
(Jim, Male)
```

```
SBP(MEAN, STD, GENDER)
(150, 20, Male)
(130, 25, Female)
```

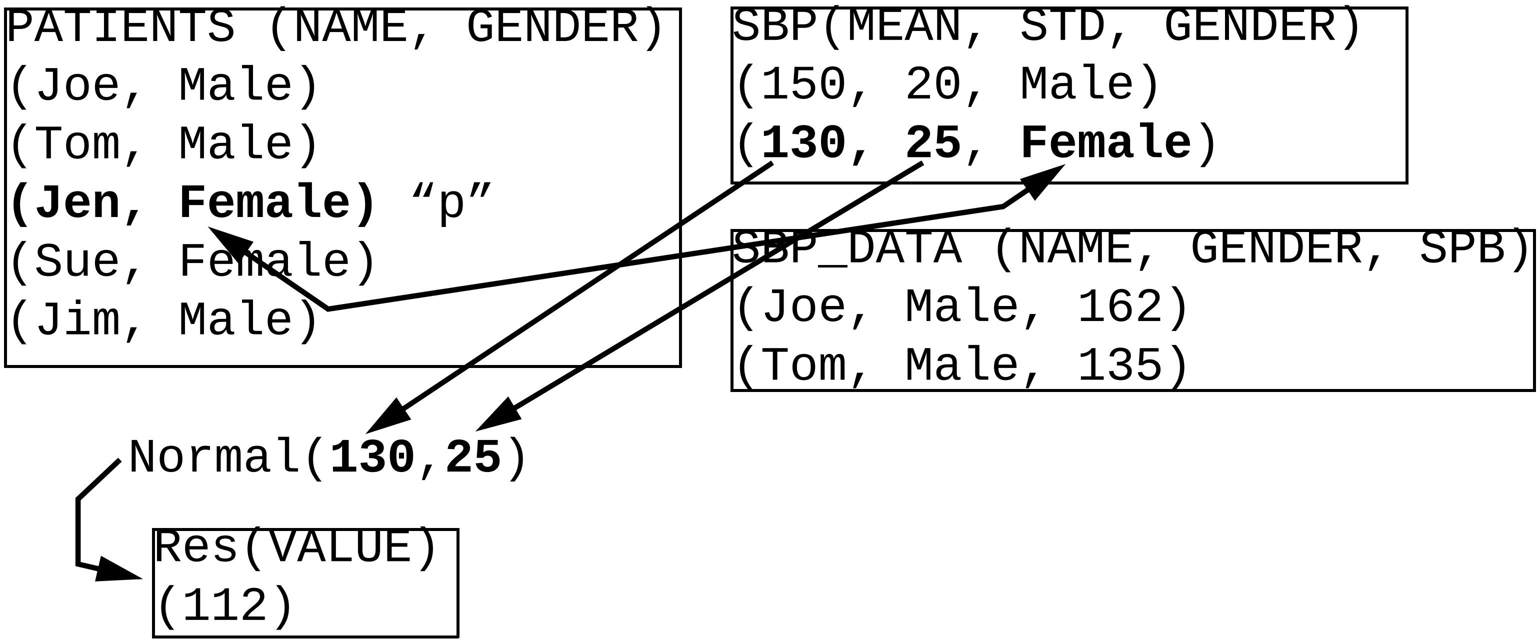
```
SBP_DATA (NAME, GENDER, SPB)
(Joe, Male, 162)
(Tom, Male, 135)
```

Normal(150,20)

```
Res(VALUE)
(135)
```

# How Does This Work?

```
CREATE TABLE SBP_DATA(NAME, GENDER, SBP) AS
FOR EACH p in PATIENTS
  WITH Res AS Normal (
    SELECT s.MEAN, s.STD
    FROM SPB s WHERE s.GENDER = p.GENDER)
  SELECT p.NAME, p.GENDER, r.VALUE
  FROM Res r
```



```
PATIENTS (NAME, GENDER)
(Joe, Male)
(Tom, Male)
(Jen, Female) “p”
(Sue, Female)
(Jim, Male)
```

```
SBP(MEAN, STD, GENDER)
(150, 20, Male)
(130, 25, Female)
```

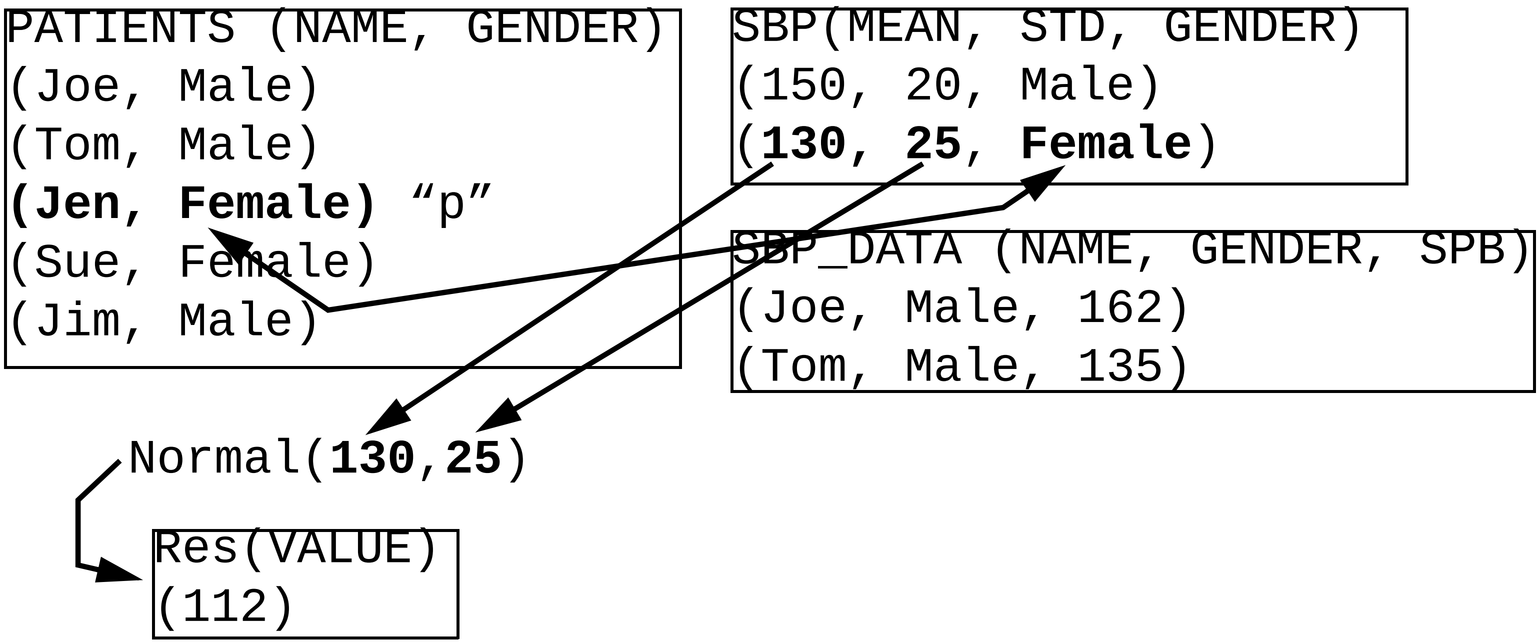
```
SBP_DATA (NAME, GENDER, SPB)
(Joe, Male, 162)
(Tom, Male, 135)
```

```
Normal(130,25)
```

```
Res(VALUE)
(112)
```

# How Does This Work?

```
CREATE TABLE SBP_DATA(NAME, GENDER, SBP) AS
FOR EACH p in PATIENTS
  WITH Res AS Normal (
    SELECT s.MEAN, s.STD
    FROM SPB s WHERE s.GENDER = p.GENDER)
  SELECT p.NAME, p.GENDER, r.VALUE
  FROM Res r
```



PATIENTS (NAME, GENDER)
(Joe, Male)
(Tom, Male)
**(Jen, Female)** “p”
(Sue, Female)
(Jim, Male)

SBP(MEAN, STD, GENDER)
(150, 20, Male)
(**130**, **25**, **Female**)

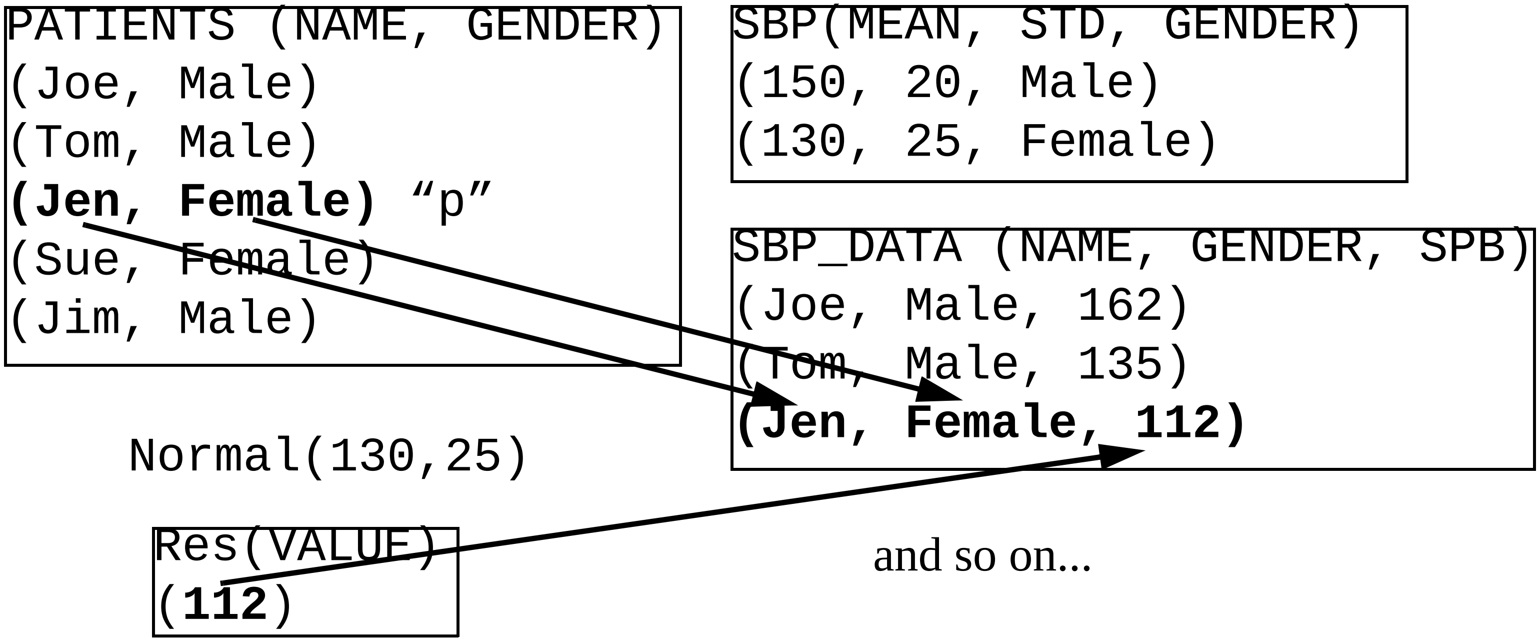
SBP_DATA (NAME, GENDER, SPB)
(Joe, Male, 162)
(Tom, Male, 135)

Normal(**130**,**25**)

Res(VALUE)
(112)

# How Does This Work?

```
CREATE TABLE SBP_DATA(NAME, GENDER, SBP) AS
FOR EACH p in PATIENTS
  WITH Res AS Normal (
    SELECT s.MEAN, s.STD
    FROM SPB s WHERE s.GENDER = p.GENDER)
  SELECT p.NAME, p.GENDER, r.VALUE
  FROM Res r
```



```
PATIENTS (NAME, GENDER)
(Joe, Male)
(Tom, Male)
(Jen, Female) “p”
(Sue, Female)
(Jim, Male)
```

```
SBP(MEAN, STD, GENDER)
(150, 20, Male)
(130, 25, Female)
```

```
SBP_DATA (NAME, GENDER, SPB)
(Joe, Male, 162)
(Tom, Male, 135)
(Jen, Female, 112)
```

```
Normal(130,25)
```

```
Res(VALUE)
(112)
```

and so on...

# Now When I Ask a Question…

- “*What is the average SBP by gender?*”
  - — Stochastic table we defined was `SBP_DATA (NAME, GENDER, SPB)`

```
SELECT GENDER, AVG (SBP) AS AVG_SBP
FROM SBP_DATA
GROUP BY GENDER
```

- We get back a *distribution* of sets of `(GENDER, AVG_SBP)` records

# More Complicated Models

- Previous allows (for example) table-valued RVs
- But Markov chains are easy in MCDB, so Bayesian ML easy
- Here’s a silly Markov chain. We have:
  - `PERSON (pname)`
  - `PATH (fromCity, toCity, prob)`
  - `RESTAURANT (city, rname, prob)`

# Markov Chain Simulation

- To select an initial starting position for each person:

```
CREATE TABLE POSITION[0] (pname, city) AS
FOR EACH p IN PERSON
  WITH City AS DiscreteChoice (
    SELECT r DISTINCT toCity
    FROM PATH)
  SELECT p.pname, City.value
  FROM City
```

# Markov Chain Simulation

- And then randomly select a restaurant:

```
CREATE TABLE VISITED[i] (pname, rname) AS
FOR EACH p IN PERSON
  WITH Visit AS Categorical (
    SELECT r.rname, r.prob
    FROM RESTAURANT r, POSITION[i] l
    WHERE r.city = l.city AND l.pname = p.pname)
  SELECT p.pname, Visit.val
  FROM Visit
```

# Markov Chain Simulation

- And transition the person:

```
CREATE TABLE POSITION[i] (pname, city) AS
FOR EACH p IN PERSON
  WITH Next AS Categorical (
    SELECT PATH.tocity, PATH.prob
    FROM PATH, POSITION[i - 1] l
    WHERE PATH.fromcity = l.city AND l.pname = p.pname)
  SELECT p.pname, Next.val
  FROM Next
```

# Markov Chain Simulation

- And transition the person:

```
CREATE TABLE POSITION[i] (pname, city) AS
FOR EACH p IN PERSON
  WITH Next AS Categorical (
    SELECT PATH.tocity, PATH.prob
    FROM PATH, POSITION[i - 1] l
    WHERE PATH.fromcity = l.city AND l.pname = p.pname)
  SELECT p.pname, Next.val
  FROM Next
```

- Fully spec’ed a Markov chain!

# Markov Chain Simulation

- To ask “How many people visit each restaurant in Houston”:

```
SELECT v.rname, COUNT(*) AS cnt
FROM VISITED AS v, RESTAURANT AS r
WHERE v.rname = r.rname AND r.city = “Houston”
```

- We get back a *distribution* of sets of `(rname, cnt)` records

# Closing Remarks

- How is this related to biomedical informatics?
  - — MCDB makes a lot of sense as a CDW platform
  - — Riddled with missing data, integration error
- An expert in stats/ML defines the models
  - — MCDB stores them just like data
- Once stored, no difference between models and data
  - — Except queries that touch models return a *distribution* of results
  - — Non-experts in stats/ML (programmers, clinicians) use the models transparently

# MCDB/SimSQL Is Open Source

- Download today!

`cmj4.web.rice.edu/SimSQL`

# Questions?